*ITEA Journal* 2009; 30: 99–110

# Embedded Instrumentation Systems Architecture

**Nikita A. Visnevski, Ph.D.**
Global Research Center of the General Electric Corporation,
1 Research Circle, Niskayuna, New York

*The objective of the Embedded Instrumentation Systems Architecture (EISA) initiative is to develop a comprehensive methodology for large-scale, nonintrusive, flexible data collection for test and evaluation needs. These needs include system-level developmental, operational, and continuous test and evaluation. The architecture can also be useful in monitoring, diagnostics, and health management, as well as protection in control applications. This article explains how EISA offers a metadata-driven methodology for heterogeneous data collection and aggregation in a synchronized and time-correlated fashion. It also describes how EISA supports real-time instrumentation and sensor management as well as virtual (synthetic) instrumentation. Finally, it addresses EISA scalability to System of Systems and/or Family of Systems embedded instrumentation applications.*

**Key words:** Embedded instrumentation; families of systems; IEEE 1451; nonintrusive instrumentation; smart sensor technology; synthetic instrumentation; systems architecture; systems of systems; virtual sensors.

The objective of the Embedded Instrumentation Systems Architecture (EISA) initiative is to develop a comprehensive methodology for large-scale, nonintrusive, flexible data collection for U.S. Department of Defense (DoD) Test and Evaluation (T&E) needs. These needs include developmental, operational, and continuous T&E of military weapons and equipment to ensure their operational readiness both at the test ranges and during the entire life cycle of the assets.

Even though the DoD is the driving force behind this architecture, commercial, industrial, and scientific communities can also benefit from such a comprehensive nonintrusive instrumentation and data acquisition methodology in the areas of system testing, monitoring, diagnostics and health management, as well as protection in control.

This article extends the previous work presented by Visnevski (2008). It explains how EISA offers a metadata-driven methodology for heterogeneous data collection and aggregation in a synchronized and time-correlated fashion. It also describes how EISA supports real-time instrumentation and sensor management as well as virtual (synthetic) instrumentation. Finally, it addresses EISA scalability to System of Systems (SoS) and/or Family of Systems (FoS) embedded instrumentation applications.

The rest of the article is organized as follows. The next section describes the EISA from the operational and system standpoints. The third section describes the demonstration platform used in the effort to validate the first reference instantiation of the architecture. It included instrumenting and testing a high-temperature superconducting power generator. The final section offers some concluding remarks and describes future research efforts to refine the EISA and expand it beyond the scope of a single large-scale system into a SoS domain.

## EISA description

The full-scale EISA description document (Visnevski 2007) contains detailed descriptions of operational, systems, and technical models of the architecture. In this article, we only highlight what we consider to be most important to the wide scientific and industrial audience aspects of the architecture.

### *EISA system-level description*

EISA follows a conventional embedded instrumentation system model shown in *Figure 1*. In a typical embedded instrumentation system, only a minimal subset of sensors and instrument components reside onboard of the system under test. This is dictated by the limits of computational resources and available

# Report Documentation Page

*Form Approved*
*OMB No. 0704-0188*

| | | |
|---|---|---|
| 1. REPORT DATE **MAR 2009** | 2. REPORT TYPE | 3. DATES COVERED **00-00-2009 to 00-00-2009** |
| 4. TITLE AND SUBTITLE **Embedded Instrumentation Systems Architecture** | | 5a. CONTRACT NUMBER |
| | | 5b. GRANT NUMBER |
| | | 5c. PROGRAM ELEMENT NUMBER |
| 6. AUTHOR(S) | | 5d. PROJECT NUMBER |
| | | 5e. TASK NUMBER |
| | | 5f. WORK UNIT NUMBER |
| 7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **General Electric Corporation,Global Research Center,1 Research Circle,Niskayuna,NY,12309** | | 8. PERFORMING ORGANIZATION REPORT NUMBER |
| 9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES) | | 10. SPONSOR/MONITOR'S ACRONYM(S) |
| | | 11. SPONSOR/MONITOR'S REPORT NUMBER(S) |

12. DISTRIBUTION/AVAILABILITY STATEMENT
**Approved for public release; distribution unlimited**

13. SUPPLEMENTARY NOTES

14. ABSTRACT

15. SUBJECT TERMS

| 16. SECURITY CLASSIFICATION OF: | | | 17. LIMITATION OF ABSTRACT | 18. NUMBER OF PAGES | 19a. NAME OF RESPONSIBLE PERSON |
|---|---|---|---|---|---|
| a. REPORT **unclassified** | b. ABSTRACT **unclassified** | c. THIS PAGE **unclassified** | **Same as Report (SAR)** | **12** | |

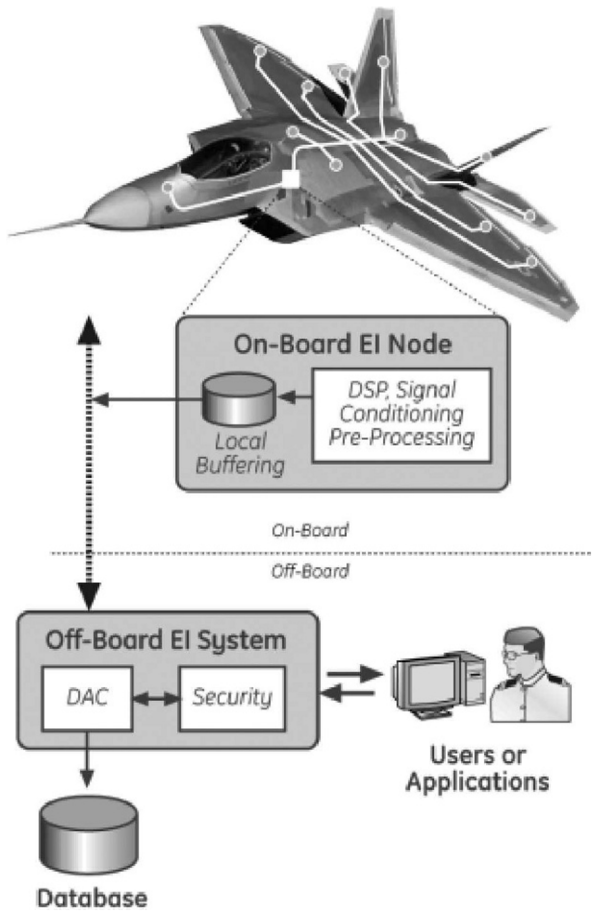

Figure 1. Typical structure of an Embedded Instrumentation System.

power onboard most mobile platforms. Instrumentation data are then preprocessed and streamed or transmitted on-demand through a telemetry network to an offboard instrumentation and data collection module. EISA follows this fundamental model. Embedded sensors and actuators of the system under test are linked to an onboard system node that is responsible for physical sensor management and initial data acquisition and post-processing activities. Then the data are transmitted over a telemetry network to the offboard data acquisition and aggregation module.

*Figure 2* presents a high-level structural decomposition of the EISA-based system that is derived from the principles illustrated in *Figure 1*. Here, onboard components are represented by the Embedded Instrumentation Nodes (EI Nodes). The job of these nodes is to aggregate data from a variety of embedded legacy of smart sensors, manage these sensors, preprocess the data, and transmit the data to an offboard system over an external telemetry network. The offboard system, called the Data Acquisition and Control (DAC) unit, is responsible for aggregating sensor data from multiple EI Nodes, implementing data arbitration between various data customers and storage systems, and maintaining proper levels of security and access control to the measurement data.

*Figure 3* illustrates the detailed architecture of the EI Node, and *Figure 4* shows the architecture of the DAC unit. One of the main characteristics of an EI Node in *Figure 3* is the fact that it takes advantage of the IEEE 1451 family of standards for smart sensors and transducers (Lee and Song 2003; Song and Lee 2006). The EI Node uses the IEEE 1451.X system model and extends it to support legacy instruments and sensors that are not IEEE 1451–compliant. This support is realized by a legacy transducer layer that can be custom built to any legacy sensor, transducer, or instrument. EI Nodes also take advantage of the IEEE 1451.1 concept of function blocks to implement sophisticated sensor- and system-level metadata management modules as well as modules enabling synthetic or virtual sensors and instruments.

The DAC unit shown in *Figure 4* enables sophisticated data and metadata management, system and sensor control mechanisms, distributed data storage, distributed data access, and user interface management and provides links with external applications and sensor data customers such as DoD Test and Training Enabling Architecture (TENA).

EISA brings the following advantages. It is fully metadata-driven in a sense that sensor data are accompanied by the metadata, which contains configuration and calibration parameters explaining exactly when and how the data were measured and what state the sensor was in during this time. System-level metadata describe what state the system was in during data gathering, which is critical if test results must be reproducible.

EISA enables data aggregation from a large network of heterogeneous sensors in a time synchronized and correlated fashion. EISA also supports smart sensor technology such as plug and play and sensor discovery mechanisms, directly supporting sensors compatible with the IEEE 1451.X family of standards (Lee and Song 2003; Song and Lee 2006).

Finally, EISA enables sophisticated support of virtual (synthetic) sensors. These are virtual data collection points for which a physical sensor does not exist (harsh environment or high cost prevents system implementers from using a physical sensor). In this case, physics-based modeling and simulation can be used to derive data of interest from other physical or synthetic sources in real time.

The process of Structured Analysis (Bienvenu, Shin, and Levis 2000; Levis and Wagenhals 2000; Wagenhals et al. 2000) was used to develop the EISA

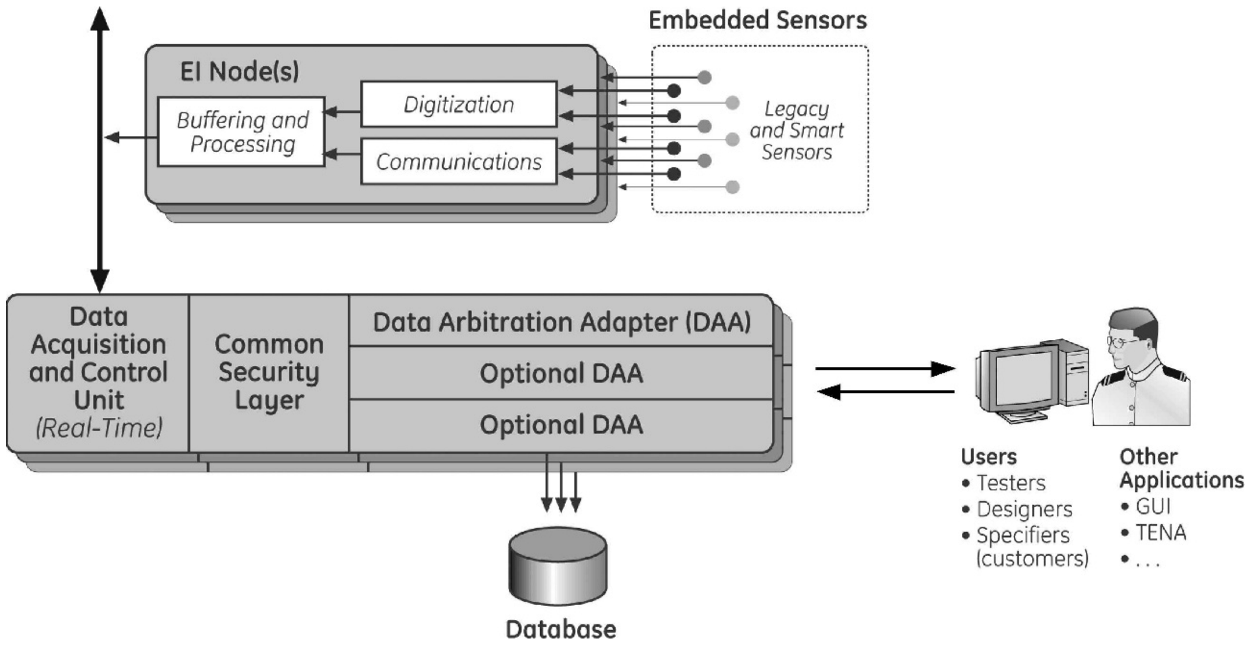

*Figure 2. High-level structural decomposition of the Embedded Instrumentation Systems Architecture–based Embedded Instrumentation System.*

architecture. At the high level, this process is an iterative development process that is also commonly referred to as a spiral development process. Systems architects typically use a variety of system modeling languages and tools (e.g., ANSI/IEEE Std. 1471–2000; Draft Federal Information Processing Std. Pub 83; IEEE Std. 1320.1 1998; Maier and Rechtin 2002; Object Management Group 2005) to capture the needed system models. Once systems architectures are developed, they can be documented in a series of

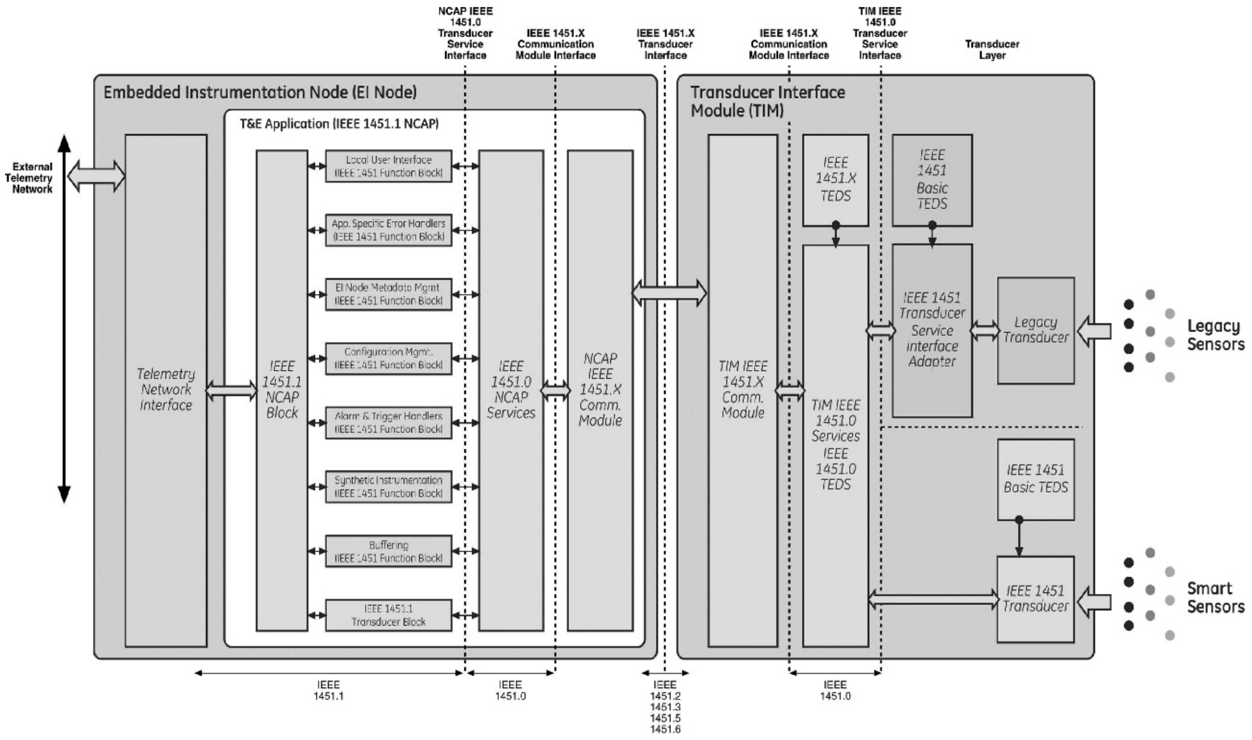
NCAP IEEE 1451.0 Transducer Service Interface
IEEE 1451.X Communication Module Interface
IEEE 1451.X Transducer Interface
IEEE 1451.X Communication Module Interface
TIM IEEE 1451.0 Transducer Service Interface
Transducer Layer
Embedded Instrumentation Node (EI Node)
T&E Application (IEEE 1451.1 NCAP)
External Telemetry Network
Telemetry Network Interface
IEEE 1451.1 NCAP Block
Local User Interface (IEEE 1451 Function Block)
App. Specific Error Handlers (IEEE 1451 Function Block)
EI Node Metadata Mgmt. (IEEE 1451 Function Block)
Configuration Mgmt. (IEEE 1451 Function Block)
Alarm & Trigger Handlers (IEEE 1451 Function Block)
Synthetic Instrumentation (IEEE 1451 Function Block)
Buffering (IEEE 1451 Function Block)
IEEE 1451.1 Transducer Block
IEEE 1451.0 NCAP Services
NCAP IEEE 1451.X Comm. Module
Transducer Interface Module (TIM)
TIM IEEE 1451.X Comm. Module
IEEE 1451.X TEDS
TIM IEEE 1451.0 Services IEEE 1451.0 TEDS
IEEE 1451 Basic TEDS
IEEE 1451 Transducer Service interface Adapter
Legacy Transducer
Legacy Sensors
IEEE 1451 Basic TEDS
IEEE 1451 Transducer
Smart Sensors
IEEE 1451.1
IEEE 1451.0
IEEE 1451.2 1451.3 1451.5 1451.6
IEEE 1451.0

*Figure 3. Embedded Instrumentation Node component of the Embedded Instrumentation Systems Architecture.*

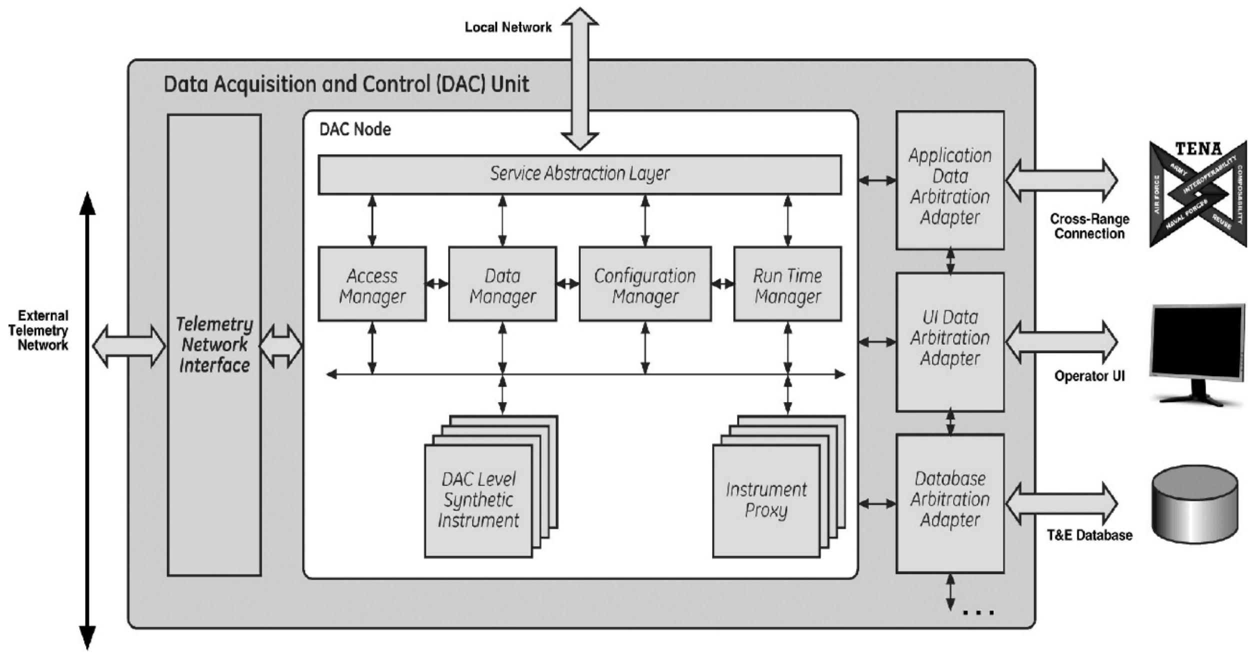
Local Network
Data Acquisition and Control (DAC) Unit
DAC Node
Service Abstraction Layer
Access Manager
Data Manager
Configuration Manager
Run Time Manager
DAC Level Synthetic Instrument
Instrument Proxy
Telemetry Network Interface
External Telemetry Network
Application Data Arbitration Adapter
UI Data Arbitration Adapter
Database Arbitration Adapter
Cross-Range Connection
TENA
Operator UI
T&E Database
...

*Figure 4. Data Acquisition and Control Unit of the Embedded Instrumentation Systems Architecture.*

graphical or textual artifacts according to an architecture framework most suitable for the application domain of the system. We used DoD Architecture Framework version 1.0 (DoD Architecture Framework Working Group 2004, deskbook, volumes I and II; Wisnosky 2005).

## EISA data model description

*Data acquisition and control unit.* The DAC unit class hierarchy shown in *Figure 5* describes the classes that are used to instantiate objects in the DAC unit software for particular applications. In this sense, it is a generic description of DAC unit software architecture. When the software for a DAC unit is instantiated for a particular application, this model determines the logical structure of software. Lower levels in the hierarchy inherit properties from those higher in the hierarchy, as indicated by the nature of the links.

The *EISA_Entity* is the top-level class that describes the system-level architecture (e.g., external interface types, number and perhaps types of DAC nodes, number and perhaps types of EI Nodes, their configuration, etc.). The *EISA_Service_Bridge* provides the software architectures suitable for interfacing with various network services (TENA services, Automated Test Markup Language, Sensor Markup Language [SML], Time Service, etc.). The *EISA_ExecutionManager* provides the software structures for managing the uploading of test plans, configuring and calibrating the system, executing tests, and transmitting the data to appropriate databases, while maintaining appropriate security access levels. The DAC portion of the associated class hierarchy is shown in lower levels of the central part of *Figure 5*. Finally the *EISA_Component* subclass provides the architectural components that correspond to and interface with key hardware functions of the DAC. These include, for instance, the software interfaces to the DAC node hardware and the generic EISA interfaces to external services and to the EI Nodes with which it is associated (termed "adapter classes" in current design practice).

Within the *EISA_ServiceBridge* class, several service subclasses are called out, reflecting the use of the Service Oriented Architecture concept in this design. The design pattern supporting these external services is normally termed a "bridge." The *TimeServiceBridge* class reflects the client interface software required for one or more time services; the *SecurityServiceBridge* reflects the interface software required to maintain security access to data and to the EISA system itself; the *DatabaseServiceBridge* reflects subscriptions of the EISA system to one or more databases (e.g., test plan database, runtime data repository, historical data repository, etc.). The *MarkupLanguageBridge* class refers to software used to interpret and exchange information in various markup languages, many of which are themselves the implementations of various standards (Automated Test Markup Language as part of IEEE1671, SML as part of IEEE of IEEE1641, etc.). These standards are being rationalized across services by the ARGCS (military) and IEEE1671 (commercial) working groups and others. Many of

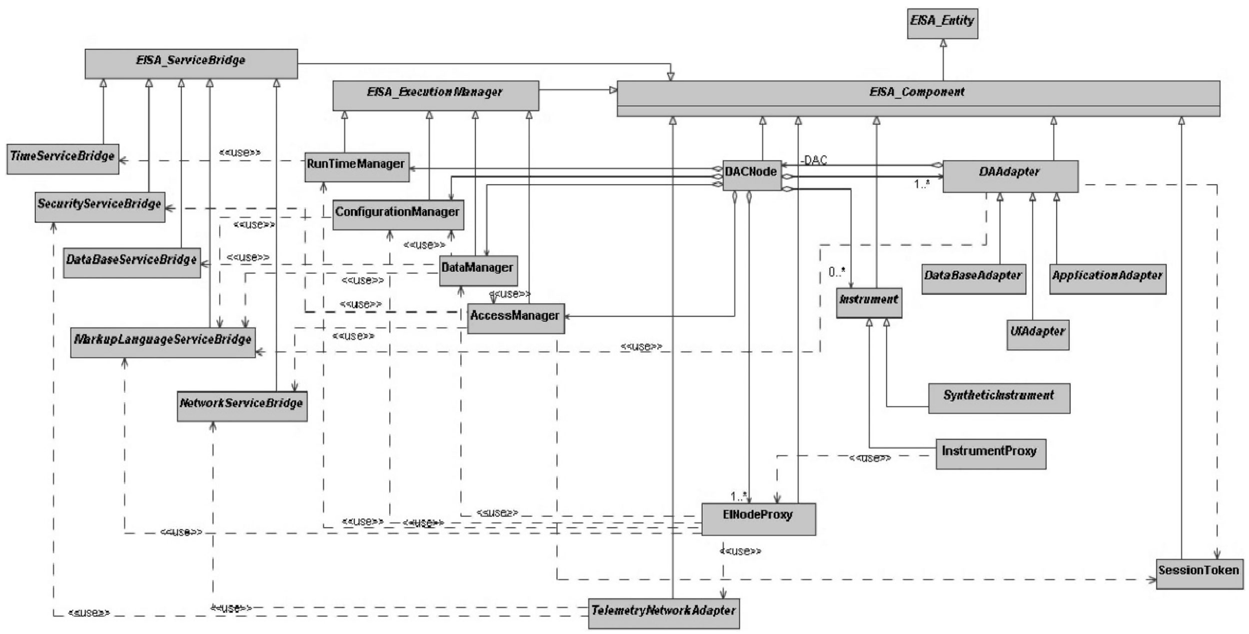

*Figure 5. Unified Modeling Language (UML) model of the Data Acquisition and Control Unit of the Embedded Instrumentation Systems Architecture.*

these rest on application-specific extensions of XML to provide common languages with which to transmit and understand various aspects of signals, instruments, and testing. The *NetworkServiceBridge* class relates to traditional network connectivity services and network error handling during service interruptions.

Extending the *EISA_ExecutionManager* class are several "manager" classes that are concerned with coordinating test execution. The *RuntimeManager* class is concerned with managing the process of embedded system testing itself (e.g., executing test scripts). The *ConfigurationManager* class is concerned with assuring that EISA is properly configured to execute a particular test and that equipment is ready when needed. *DataManagement* is concerned with proper routing of data, which typically will be streamed into external data files from the EINode equipment via the DAC; data routing may change occasionally during different test steps. *AccessManager* assures proper access to the DAC node and to data before, during, and after testing—note that test personnel may often require user interface access to the DAC node to confirm test readiness or to synchronize testing with nonmeasured phenomena, such as the state of mechanical equipment or readiness of test team personnel.

Extending the *EISA_Component* class are primarily the *DACNode* and *DAAdapter* classes. The *DACNode* class supports information and operations specific to a DAC Node instance (e.g., EINodes to which it is associated, local hardware/software interface methods, etc.), whereas the *DAAdapter* class supports high speed data routing to a local data repository (often accessed via backplane or a local area network) and coordination of one DAC unit with another, if required. The "Instrument" abstract class incorporates physical instruments viewed at the DAC level, as well as synthetic instruments realized at the DAC level (*DACSynthetic-Instument*) and a proxy class for instruments realized at the EINode level (*EIInstrumentProxy*), and synthetic instruments realized at the EI Node level (*EISynthet-icInstrumentProxy*), all of which may be accessed via the DAC in some cases.

The *DAAdapter* class is extended by the *Database Adapter*, *UIAdapter*, and *ApplicationAdapter* subclasses. These reflect the specific data processing needs of a particular application, including data routing during review or preview functions (*DatabaseAdapter*), different ways of viewing the test data (*UIAdapter*), and application-specific data processing (*ApplicationAdapter*).

Finally, the *TelemetryNetworkAdapter* and *SessionTo-ken* classes provide natural extension details of the *NetworkServiceBridge* class to support secure wireless connectivity between EINodes and DAC Nodes in this project using the iNET wireless telemetry network architecture.

*EI node.* The "EI Node" class structure shown in *Figure 6* takes its higher levels from the *EISA_Entity* and *EISA_Component*, as previously described. It also incorporates the *Instrument* capability and *EISynthet-*

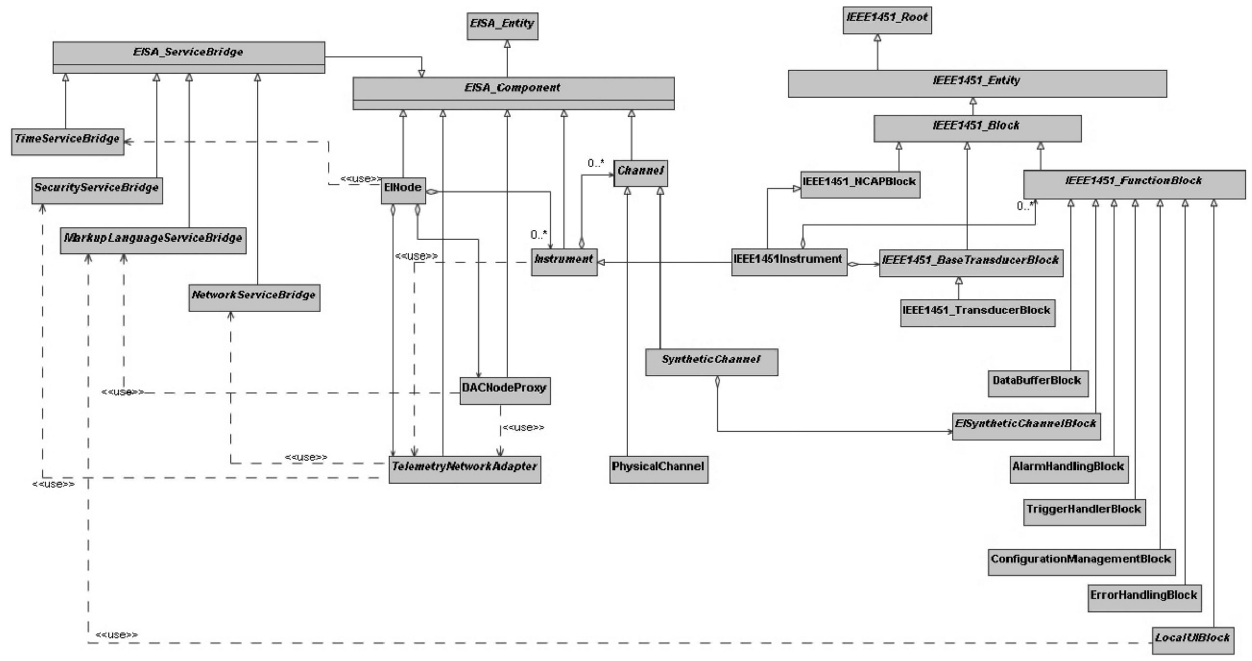

*Figure 6. Unified Modeling Language (UML) model of the Embedded Instrumentation Node component of the Embedded Instrumentation Systems Architecture.*

*icInstrument* capabilities at the *EINode* level, and 1451*Instrument* capabilities, which are standardized instrument templates supported via the IEEE1451 standard. The EI Node also specializes the *TelemetryNetworkAdapter* class and uses the *DACNodeProxy* class for secure wireless communications with its associated DAC Node. It uses a subset of the same services supported at the DAC Node via the *EISA_ServiceBridge* class, as described in the preceding section.

The low level functions of the EI Node are implemented in accordance with the IEEE 1451 Standard, as shown at the right of this class hierarchy under the *IEEE1451_Root* class. This class hierarchy is adopted here in its entirety and is identical to the class hierarchy that has been developed by the IEEE1451 Standards Committee (Lee and Song 2003). Because this hierarchy has been fully documented, only its specialization to the EISA EI Node application is discussed here. Application-specific aspects of the IEEE 1451 Standard are implemented in the *IEEE1451_NCAPBlock*, a dedicated but customizable function block within the *IEEE1451_FunctionBlock* class; the *IEEE1451_TransducerBlock* and its base class; and a set of EI Node application-specific function blocks shown at the lower right of the diagram. Precise definitions of these are provided in the IEEE 1451 Standard, but qualitatively, the Transducer Block specifies individual sensor or actuator device properties (via the TEDS data class in the metadata model), the NCAP provides the generic aspects of an interface between the EI instrument (IEEE1451.2 in this case) and a network (in this case iNET), and the application-specific blocks characterize specific behaviors of the instrument.

In the EI Node, the types of application-specific behaviors that are needed have been grouped into the *DataBufferBlock*, providing short-term data buffering; the *EISyntheticInstrument* block, providing low-level primitive instrument capabilities (e.g., multiplication or addition of signals from different sensors); the *AlarmHandlingBlock*, providing low-level alarm-handling capabilities that may be based on simultaneous conditions on several sensors; the *TriggerHandlerBlock* responsible for low-level start/stop triggering or recording of events; the *ConfigurationManagementBlock* concerned with local EI Node configuration parameters (e.g., number of associated sensors); and the *MetaDataManagement* block, in this case concerned with the description and relationships of various sensor and actuator types, the *ErrorHandlingBlock* concerned with EI Node and device-specific error handling, and the *LocalUIBlock* concerned with local viewing of particular sensor signals, e.g., for pretest checkout purposes, at the EI Node location. In the 1451 standard, this is supported generically by the XML service, but might also reflect more specialized markup languages such as SML (e.g., viewing of different signal types).

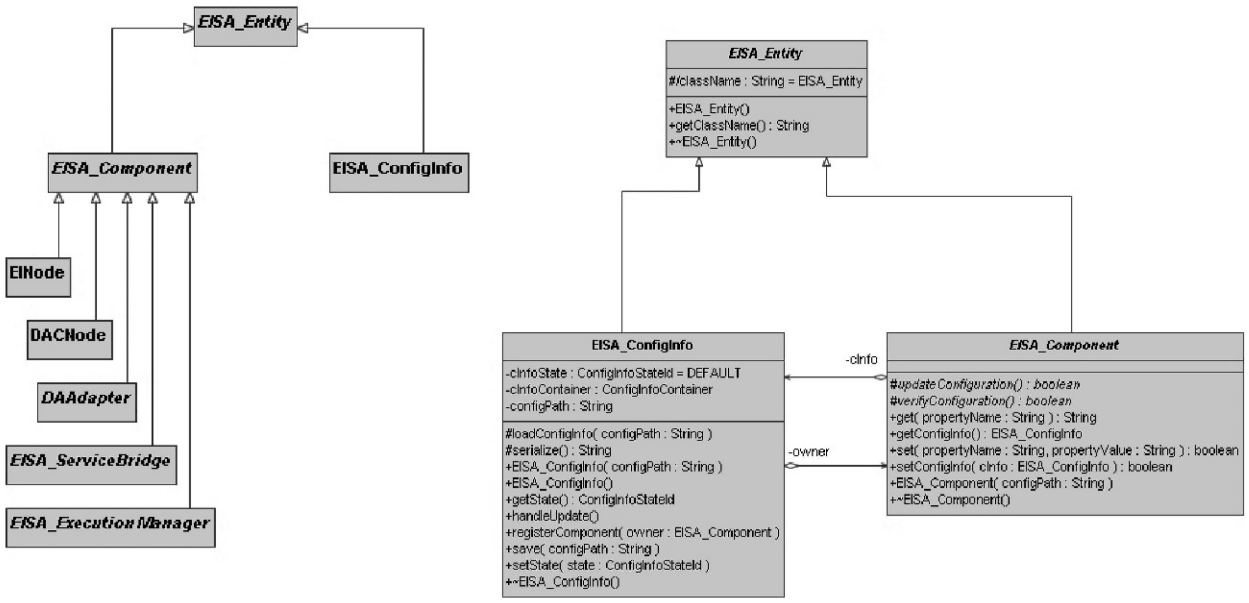
EISA_Entity
EISA_Component
EISA_ConfigInfo
EINode
DACNode
DAAdapter
EISA_ServiceBridge
EISA_Execution Manager
EISA_Entity
#/className : String = EISA_Entity
+EISA_Entity()
+getClassName() : String
+~EISA_Entity()
EISA_ConfigInfo
-cInfoState : ConfigInfoStateId = DEFAULT
-cInfoContainer : ConfigInfoContainer
-configPath : String
#loadConfigInfo( configPath : String )
#serialize() : String
+EISA_ConfigInfo( configPath : String )
+EISA_ConfigInfo()
+getState() : ConfigInfoStateId
+handleUpdate()
+registerComponent( owner : EISA_Component )
+save( configPath : String )
+setState( state : ConfigInfoStateId )
+~EISA_ConfigInfo()
-cInfo
-owner
EISA_Component
#updateConfiguration() : boolean
#verifyConfiguration() : boolean
+get( propertyName : String ) : String
+getConfigInfo() : EISA_ConfigInfo
+set( propertyName : String, propertyValue : String ) : boolean
+setConfigInfo( cInfo : EISA_ConfigInfo ) : boolean
+EISA_Component( configPath : String )
+~EISA_Component()

*Figure 7. Embedded Instrumentation Systems Architecture use of the configuration information model.*

## *EISA configuration information management scheme*

The EISA class model is defined according to the fundamental principles of object orientation in the form of a hierarchy, with the *EISA_Entity* class being the parent class of all classes in the EISA class hierarchy. There are currently two distinct EISA entities—*EISA_Component* and *EISA_ConfigInfo*. Every element of the EISA class hierarchy belongs to one of these two types. *Figure 7* illustrates the relationships between these key EISA classes.

*Figure 7* illustrates the basic EISA class hierarchy concept and demonstrates how EISA supports the concept of configuration information and metadata management. EISA exploits the duality between the concepts of configuration information and metadata in a sense that any system-level configuration information that has been successfully activated for the purpose of conducting a particular T&E activity has become the system-level metadata for the data generated during the course of this activity.

The link between an *EISA_Component* and an *EISA_ConfigInfo* is established through a circular aggregation relationship that is of fundamental importance. Each EISA component owns (aggregates an instance of) *EISA_ConfigInfo*. This ensures that each EISA component contains a unique set of configuration parameters. These parameters are loaded by the configuration manager and are validated at the time EISA component is instantiated. These parameters are accessible through "get" and "set" *EISA_Component* API methods and can be dynamically modified at runtime.

The reverse aggregation of the *EISA_Component* by an instance of *EISA_ConfigInfo* ensures that every instance of configuration information is aware of its owner and can trigger update callbacks on the owner should the content of the configuration information change. For example, a user modifies the EI Node configuration file in the configuration database. This causes an operating system callback to the configuration manager that is dispatched by the manager directly to the *EISA_ConfigInfo* class instance that was instantiated from this configuration file. The *EISA_ConfigInfo* instance then reloads itself and notifies the owner class of the change. This is supported by the "verify" and "update" API methods of the EISA component. An update method of the EI Node is called, the new configuration is verified, and is accepted or rejected based on the state of the system (see *Figure 8* for an example sequence diagram).

Note that *EISA_ConfigInfo* does not prescribe any specific configuration data fields for any specific EISA component. The idea behind it is that *EISA_ConfigInfo* is a flexible, dynamically loaded generic configuration information container that can aggregate any property-value pair style configuration information to support any EISA component in the architecture.

## EISA demonstration platform

The EISA concepts described above have been applied to data acquisition in a test of a large-scale High-Temperature Superconducting (HTS) Multi-megawatt Electric Power System (MEPS). This is a high energy density device developed at GE Research for the efficient transfer of mechanical rotation energy of a turbine into

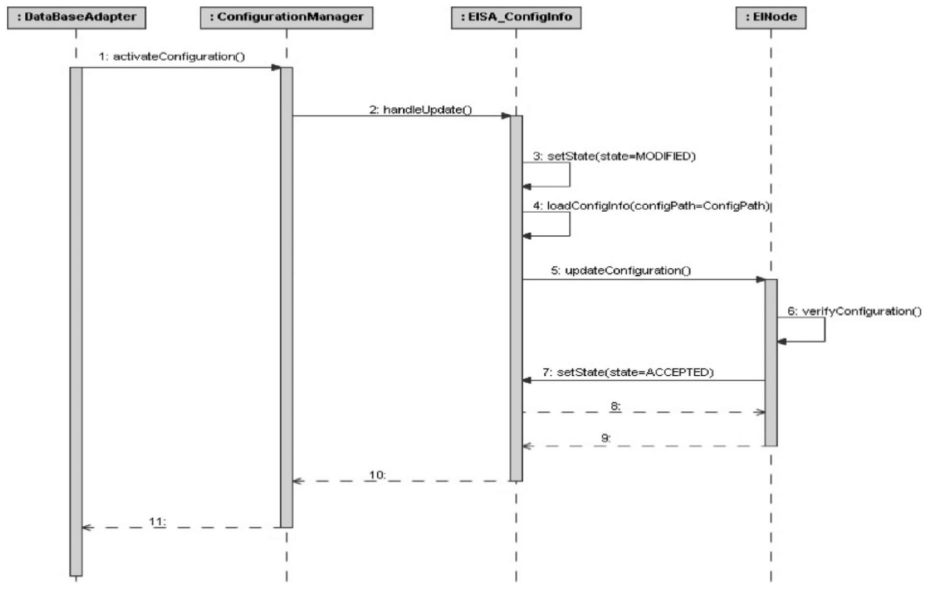

*Figure 8. An example sequence of Embedded Instrumentation Systems Architecture configuration information use.*

electric power at very high efficiency levels. The EISA implementation to support the test of this generator is described in *Figure 9*. The goal of this data acquisition system was to obtain high-quality, correlated, and time stamped test data for post-test analysis.

In addition to measuring physical sensor data, a number of synthetic data points and virtual instruments were implemented. One such set up is shown in *Figure 10*. It illustrates how synthetic calculation of losses enabled high-quality expected generator output power estimation that could be compared with the real output power measurements for quality assurance purposes.

EISA support of MEPS greatly simplified T&E of this system when compared to traditional data acquisition methods. All test data were time synchronized, which made it easy to analyze it after the test. Synthetic sensor values were calculated at run time and displayed alongside with the physical sensor data on a single unified and configurable user interface for monitoring and diagnostics purposes. Finally, reconfiguration of the test setup was very easy as it only involved appropriate modifications of the system level metadata that was dynamically reloaded by the EISA system, enabling flexible run-time reconfiguration. The rest of this section describes this experiment in more details.

## Multi-megawatt electric power generation test platform

The MEPS was chosen as the demonstration platform for the EISA architecture because of the complexity and information-intensive instrumentation requirements for the system. MEPS is an HTS generator that can be used for a variety of applications that require high-density power generation at high rotational speeds. Mobile radar, pulsed weapons, and grid power generation are among the applications for which HTS generators are being considered for use. HTS generators are more efficient than other electric power generators because of the use of superconducting coils in the windings. The HTS generator is smaller and lighter than traditional power generators because the HTS design utilizes an air core structure that eliminates the need for large quantities of steel that other electric power generators require.

Eight instrumentation systems are used to acquire data from sensors and transmit information to legacy instruments developed by different vendors. The instruments that are supported by EISA for this demonstration include:

1. an instrument for data acquisition from flow meters, pressure gauges, thermocouples, RTDs, vacuum indicators, and pyrometers;
2. an instrument for data acquisition from vibration sensors;
3. an instrument for data acquisition from pickup coils;
4. an instrument for data acquisition from voltage sensors, current sensors, torque meter, and tachometer;
5. a web camera for video data collection.

There are approximately 175 sensors in the system. The complexity of the data acquisition system is the cause of several challenges for the MEPS testers, such as:

- Each instrument time stamps the data with a separate time stamp. If a fault occurs during

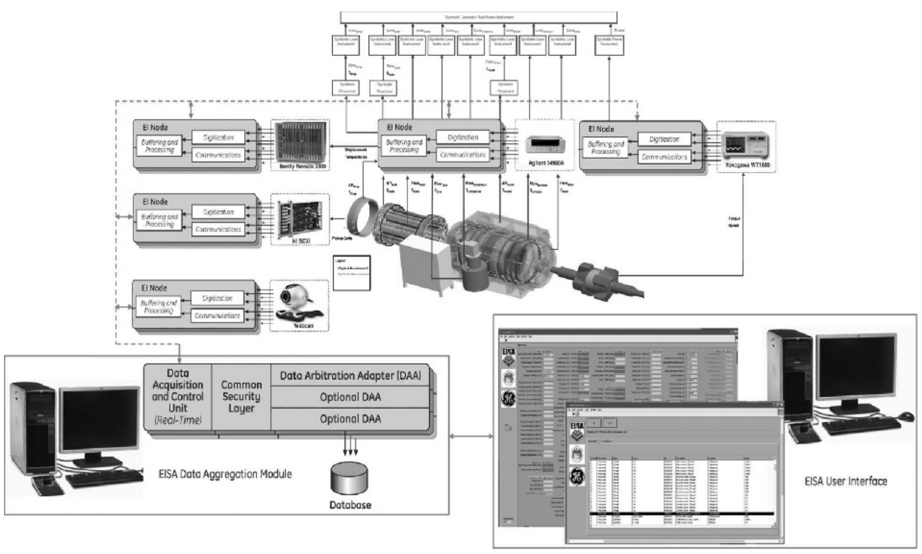

*Figure 9. Embedded Instrumentation Systems Architecture Implementation of the Multi-megawatt Electric Power System Demonstration System.*

testing, it is difficult to reconstruct event data at the time of the fault from different instruments because the data is not synchronized to a global time stamp.

- Legacy software often lacks flexibility to program complex equations and model-based estimates of synthetic measurements that are deduced from physical measurements.
- Each instrument requires a separate calibration and configuration process for the attached sensors.
- Visual data are dispersed across multiple GUIs operating on multiple processors.

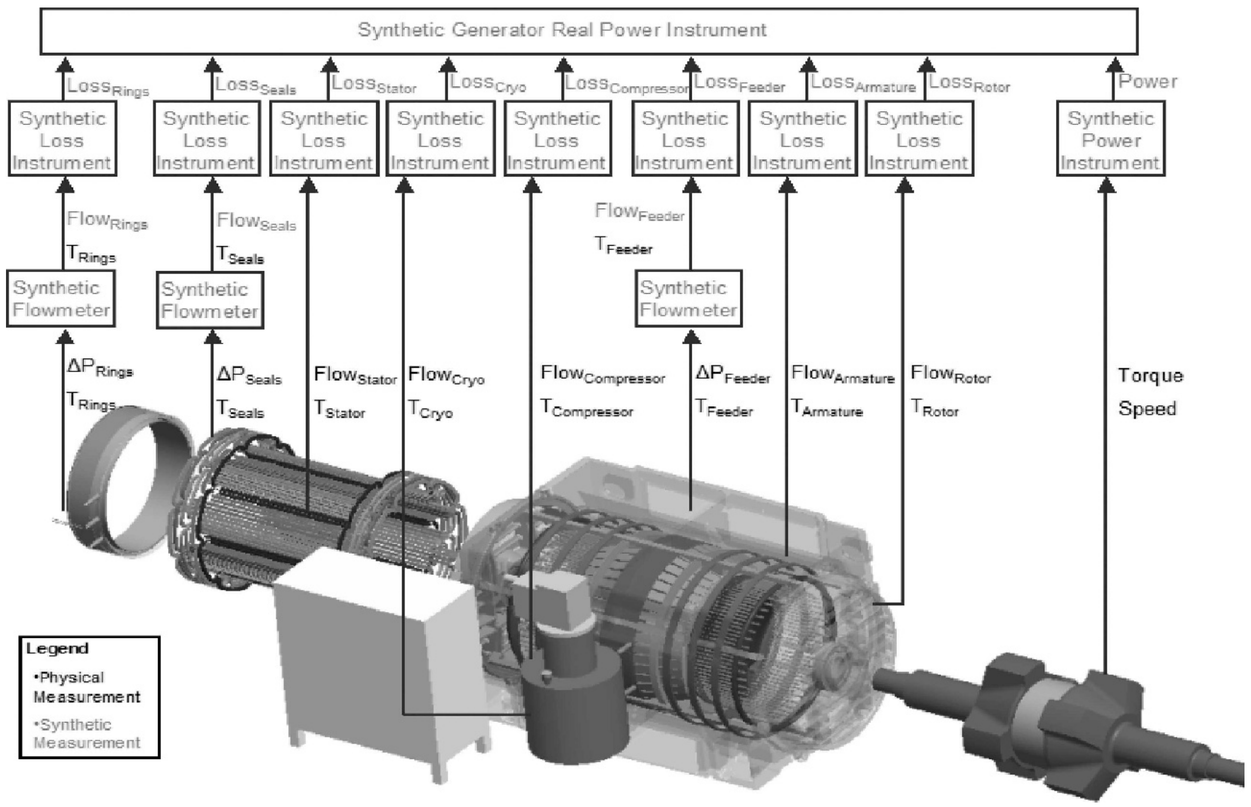
Synthetic Generator Real Power Instrument
Loss$_{Rings}$
Loss$_{Seals}$
Loss$_{Stator}$
Loss$_{Cryo}$
Loss$_{Compressor}$
Loss$_{Feeder}$
Loss$_{Armature}$
Loss$_{Rotor}$
Power
Synthetic Loss Instrument
Synthetic Power Instrument
Flow$_{Rings}$ T$_{Rings}$
Flow$_{Seals}$ T$_{Seals}$
Flow$_{Feeder}$ T$_{Feeder}$
Synthetic Flowmeter
ΔP$_{Rings}$ T$_{Rings}$
ΔP$_{Seals}$ T$_{Seals}$
Flow$_{Stator}$ T$_{Stator}$
Flow$_{Cryo}$ T$_{Cryo}$
Flow$_{Compressor}$ T$_{Compressor}$
ΔP$_{Feeder}$ T$_{Feeder}$
Flow$_{Armature}$ T$_{Armature}$
Flow$_{Rotor}$ T$_{Rotor}$
Torque
Speed
Legend
•Physical Measurement
•Synthetic Measurement

*Figure 10. Logical flow of data from Multi-megawatt Electric Power System to synthetic instruments.*

- Each instrument utilizes its own user authentication and data control process. Developing a comprehensive data security process is difficult.

*Figure 9* shows the EISA implementation of the MEPS demonstration system. In the HTS MEPS test platform, 175 heterogeneous sensors have been embedded in a variety of the generator and test assembly components. They were wired to a set of commercial off-the-shelf data acquisition systems (Agilent, Yokogawa, Bently Nevada, National Instruments, Logitech, etc.). In this configuration, separate data acquisition systems produced data at different data rates that were not correlated, not time synchronized, were stored in different databases, and posed challenges for post-processing. The EISA-based solution involved building IEEE 1451 wrappers around the commercial data acquisition systems. This enabled continuous and integrated data and metadata aggregation for the entire test system. The test data were automatically time synchronized and stored in a single database, greatly simplifying post-test analysis.

## EISA support of synthetic instrumentation in MEPS

One of the very useful properties of EISA is its inherent ability to support real-time synthetic instrumentation. EISA enables MEPS testers to integrate synthetic instrumentation to extend the capabilities of the data acquisition systems. Synthetic instrumentation of a varying degree of complexity can be implemented to compliment and extend physical measurements acquired by existing instrumentation.

Synthetic instrumentation refers to instruments that can measure quantities through numerical processing by using input from various physical sensors. Quantities measured by synthetic instruments cannot be measured directly using physical sensors because they require the combination of inputs from multiple sensors. Synthetic instruments add modularity and scalability to the architecture at a low cost because the system hardware requires minimum modification to include additional synthetic instruments in the system.

Synthetic instruments developed for MEPS are separated into two groups. The first group of synthetic instruments will use inputs from a single EI Node to measure the virtual quantity. The second group of synthetic instruments will use inputs from multiple EI Nodes that are aggregated in the database to measure the virtual quantity. These groups of synthetic instruments are shown in *Figure 10* and described in more details below.

*Generator losses.* Generator losses occur because of the energy expended in the generator's internal resistance and to cool various components during operation. Losses are an indicator of the generator efficiency and true output power. Thermal losses can be calculated by measuring the expended energy for the various cooling loops in the system. The generator losses include:

1. stator core losses,
2. stator bar losses
3. end winding losses,
4. connection ring losses,
5. rotor losses,
6. ferrofluid losses,
7. cryostat losses,
8. core losses in laminations,
9. pump losses,
10. feeder losses.

The data acquired to calculate the delivered power come from two EI Nodes. One node calculates the drive power via torque and speed, and the second node provides data to calculate the thermal losses in the system. All thermal losses are calculated using the general formula:

$$Energy = mCp\Delta T \qquad (1)$$

Flow meters are installed in some of the cooling loops to calculate the volumetric flow rate. However, the ionized water cooling loop will not contain flow meters and will require the development of a synthetic flow meter. The delivered power at the terminals of the generator can be calculated by subtracting the sum of the system losses from the total power generated, which is equal to the product of torque and speed. *Figure 10* shows the logical flow of data from MEPS to the synthetic instruments.

*Synthetic flow meter.* Calculating flow rate for cooling loops is necessary to determine if the amount of coolant flowing through the system is sufficient. Flow meters are high cost instruments that cannot be placed at numerous points in the system. A synthetic flow meter is developed to measure flow using the differential pressure and temperature between the inlet and outlet of the cooling loops in the deionized water system.

*Delta temperature of cooling loops.* Another set of synthetic instruments is developed for thermal protection. The goal of the instruments is to measure the difference in temperature between the nominal temperature of the cooling loop and the temperature of the different components in the loop. A large difference in temperature in one section of the cooling loop can indicate a faulty cooling system that requires shutdown and maintenance.

*Electrical fault detection.* Early detection of electrical faults is important to protect the generator and shut down the system before damage occurs. Measuring current balance among the three-phase sets is important

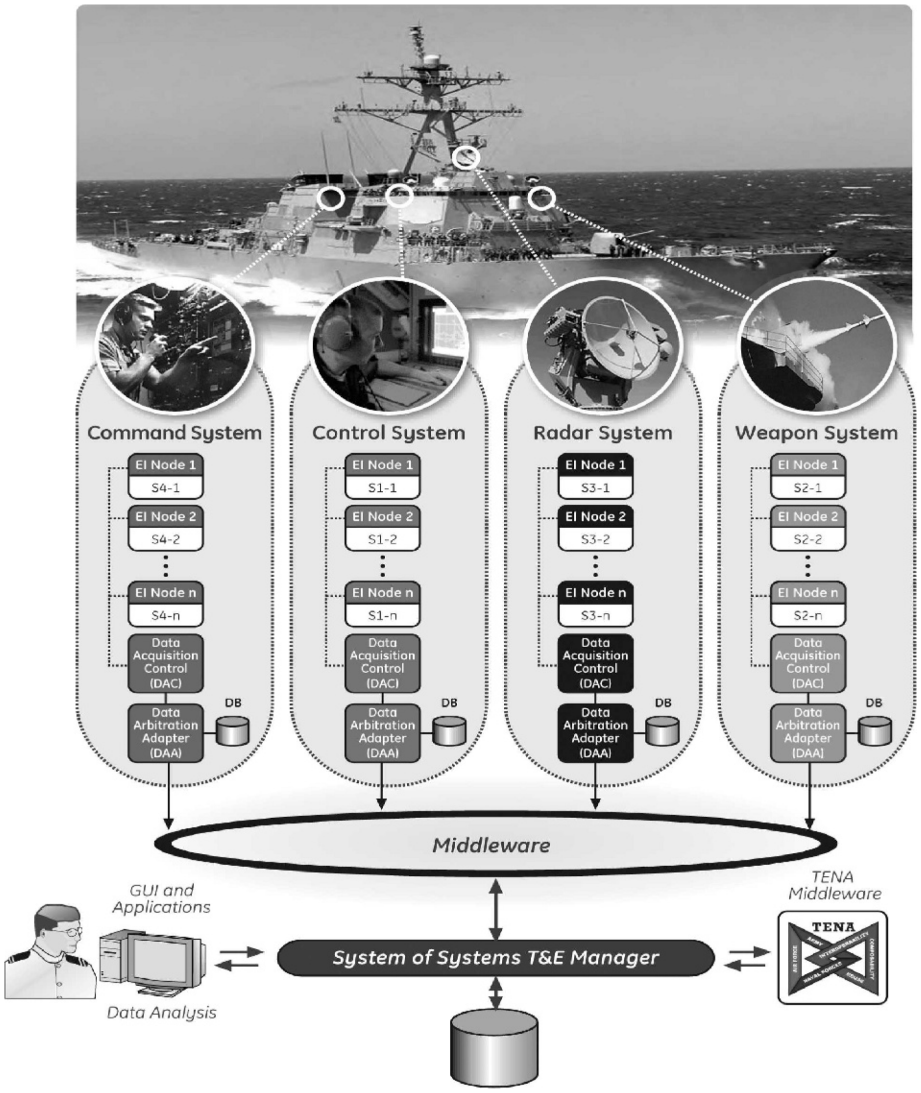

*Figure 11. Embedded Instrumentation Systems Architecture System of Systems instrumentation support.*

because current unbalance will cause rotor surface heating. Current unbalance can come from the individual differences of the phases in the generator, connections, and load banks. One cause of sudden current unbalance can come from the shorting of a resistor in the load bank, for example. A synthetic instrument is developed to detect current unbalance between the three phases. Another synthetic instrument is developed to detect voltage unbalance between the three phases. Voltage unbalance degrades the performance and shortens the lifetime of the generator because the voltage unbalance leads to current balance, which results in overheating of the generator as previously discussed.

The current balance of the phases A, B, and C can be verified by dividing the highest current magnitude among three phases by the lowest magnitude among the three phases followed by a division of the largest current phase angle by the lowest phase angle. If the ratio of the currents is close to one, the phase currents are balanced. The same procedure is utilized for monitoring three-phase voltage balance.

## Discussions and future work

In this article, we described an architecture called an Embedded Instrumentation Systems Architecture. This architecture facilitates multirate heterogeneous data acquisition for complex large-scale system T&E. It is metadata-driven in the sense that sensor and system level metadata determine automated test system configuration dynamically at run time. It supports IEEE 1451–based smart sensor technology and provides a flexible platform for enabling real-time synthetic or virtual instrumentation. This architecture could be useful in military and commercial T&E applications as well as monitoring, diagnostics, health management, and control applications.

This article also described the application of the EISA architecture in the context of testing a single large-scale system—an HTS power generator. EISA brought substantial benefits to the generator testing by enabling seamless and cohesive integrated test data aggregation and enabling real-time synthetic measurements of test points that were not directly measurable.

The future of EISA includes work to extend the architecture beyond supporting T&E of a single large-scale system to the domain of SoS testing. This involves testing of sophisticated hierarchical test subjects such as an entire ship, airplane, cluster of unmanned vehicles, etc. GE Research is currently involved in developing this SoS architecture concept and demonstrating it on a declassified version of the command and control infrastructure of the new-generation destroyer. This concept is illustrated in *Figure 11*. Because EISA is following a centralized data aggregation path, the SoS architecture development focus is on the nodes that tie together individual DAC units associated with various data aggregation systems. This architecture component is called SoS T&E Manager (*see Figure 11*). Its goal is to preserve a coordinated data acquisition strategy across heterogeneous systems in the SoS testing framework and enable flexible control of testing process automation. ❑

*NIKITA A. VISNEVSKI completed his undergraduate studies in Computer and Software Engineering at the St. Petersburg Academy of Aerospace Engineering, St. Petersburg, Russia, in 1995. He received his M.S. degree in Electrical Engineering from Texas Tech University, Lubbock, Texas, in 1997, and his Ph.D. in Electrical Engineering from McMaster University, Hamilton, Ontario, Canada, in 2005.*

*His industrial experience includes working for the Boeing Company, Ford Motor Company, and The MathWorks, Inc. Since August 2005 he has been a staff scientist and system architect at the Global Research Center of the General Electric Company. He is a member of the Signal Electronics & Embedded Systems Laboratory of the Electronics and Energy Conversion Organization. His primary research focus areas include intelligent and cognitive embedded systems as well as large-scale systems architectures. E-mail: Nikita.Visnevski@research.ge.com*

## References

ANSI/IEEE Standard 1471–2000. 2004. Recommended Practice for Architectural Description of Software-Intensive Systems.

Bienvenu, M. P., I. Shin. and A. H. Levis. 2000. C4ISR architectures: III. An object-oriented approach for architecture design. *Systems Engineering*. 3 (4): 288–312.

DoD Architecture Framework Working Group. February 2004. DoD architecture framework version 1.0: Deskbook.

DoD Architecture Framework Working Group. February 2004. DoD architecture framework version 1.0: Volume I definitions and guide-lines.

DoD Architecture Framework Working Group. February 2004. DoD architecture framework version 1.0: Volume II product descriptions.

Draft Federal Information Processing Standards Publication 183. 1993. Integration Definition For Function Modeling (IDEF0).

IEEE Std 1320.1. 1998. IEEE Standard for Functional Modeling Language–Syntax and Semantics for IDEF0.

Lee, K. and E. Y. Song. 2003. UML model for the IEEE 1451.1 standard. *Instrumentation and Measurement Technology Conference*. 2: 1587–1592.

Levis, A. H. and L. W. Wagenhals. 2000. C4ISR architectures: I. Developing a process for C4ISR architecture design. *Systems Engineering*. 3 (4): 225–247.

Maier, M. W. and E. Rechtin. 2002. *The art of systems architecting*. 2nd ed. Boca Raton, FL: CRC Press.

Object Management Group. 2005, August. Unified modeling language: Superstructure. version 2.0. Needham, MA: Object Management Group (OMG). Accessed online at http://www.omg.org/cgi-bin/doc?ptc/06-10-06.

Song, E. Y. and K. Lee. 2006. An implementation of the proposed IEEE 1451.0 and 1451.5 standards. *Sensors Applications Symposium*. Proceedings. 72–77.

Visnevski, N. A. November 2007. Embedded Instrumentation Systems Architecture description, version 1.3., Public-domain report submitted to the Navy Undersea Warfare Center.

Visnevski, N. A. 2008. Embedded Instrumentation Systems Architecture. In *Proceedings International Instrumentation and Measurement Technology Conference*. May 12–15, 2008, Victoria, British Columbia, Canada, pp. 1134–1139, IEEE.

Wagenhals, L. H., I. Shin., D. Kim. and A. H. Levis. 2000. C4ISR architectures: II. A structured analysis approach for architecture design. *Systems Engineering*. 3 (4): 248–287.

Wisnosky, D. E. 2005. *DoDAF wizdom*. Naperville, IL: Wizdom Systems, Inc.

## Acknowledgments

The author would like to thank the Test Resource Management Center (TRMC) Test and Evaluation/Science and Technology (T&E/S&T) Program for their support. This work is funded by the T&E/S&T Program through the Naval Undersea Warfare Center, Newport, Rhode Island, contract N66604-05-C-2335.